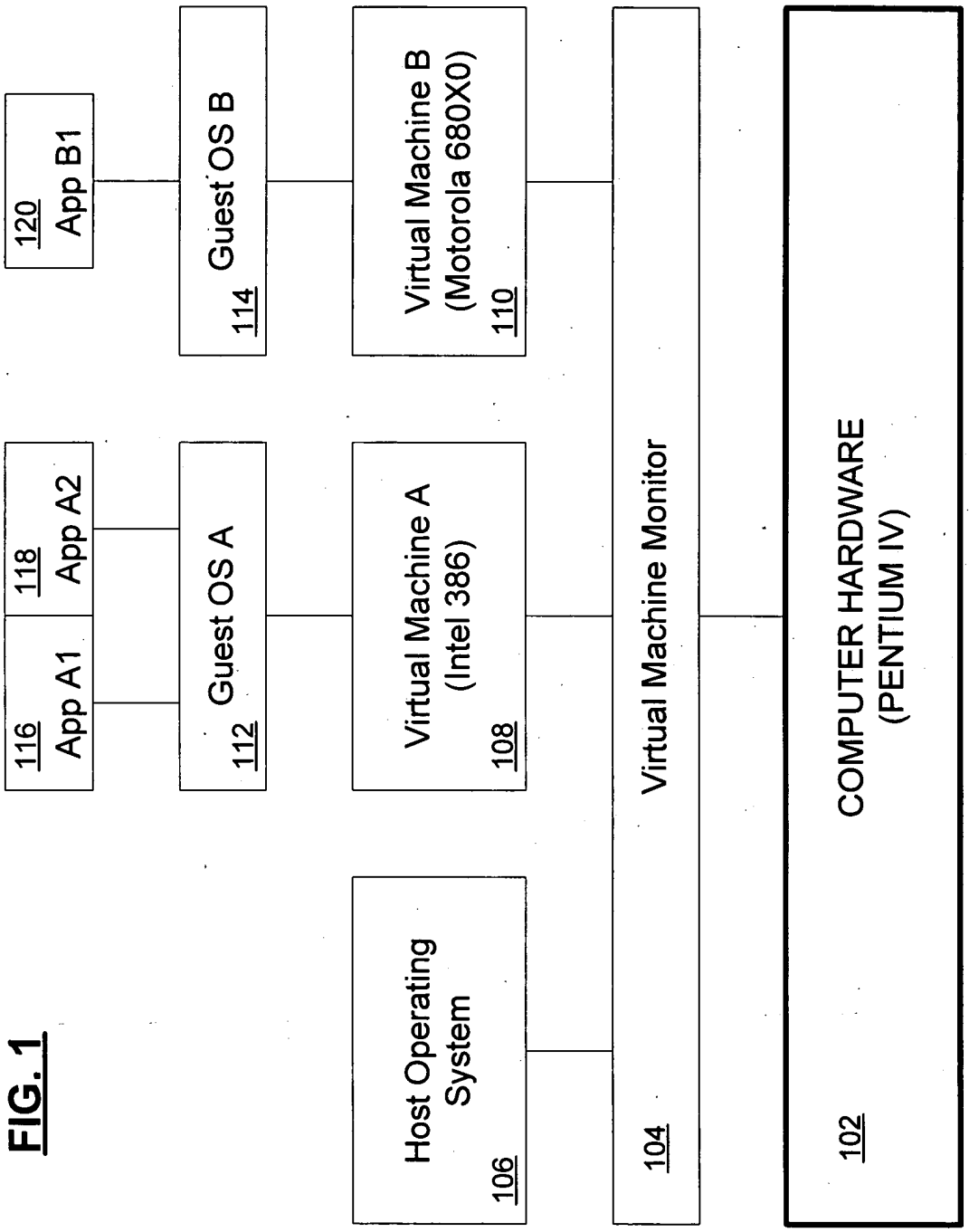
120 App B1
Guest OS B 114
Virtual Machine B (Motorola 680X0) 110
116 App A1
118 App A2
Guest OS A 112
Virtual Machine A (Intel 386) 108
Host Operating System 106
Virtual Machine Monitor 104
COMPUTER HARDWARE (PENTIUM IV) 102

FIG. 1

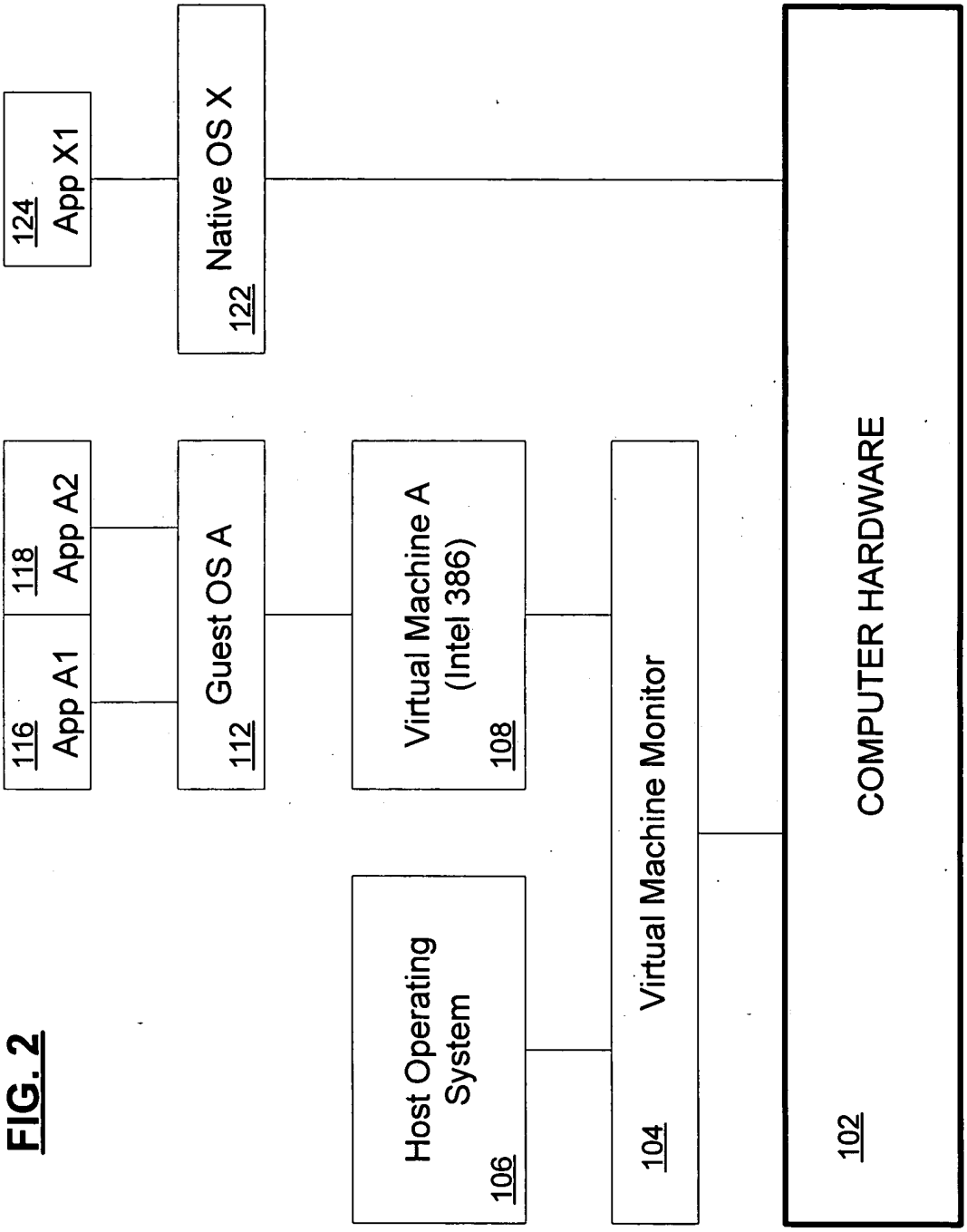
124 App X1
Native OS X 122
116 App A1
118 App A2
Guest OS A 112
Virtual Machine A (Intel 386) 108
Host Operating System 106
Virtual Machine Monitor 104
COMPUTER HARDWARE 102

FIG. 2

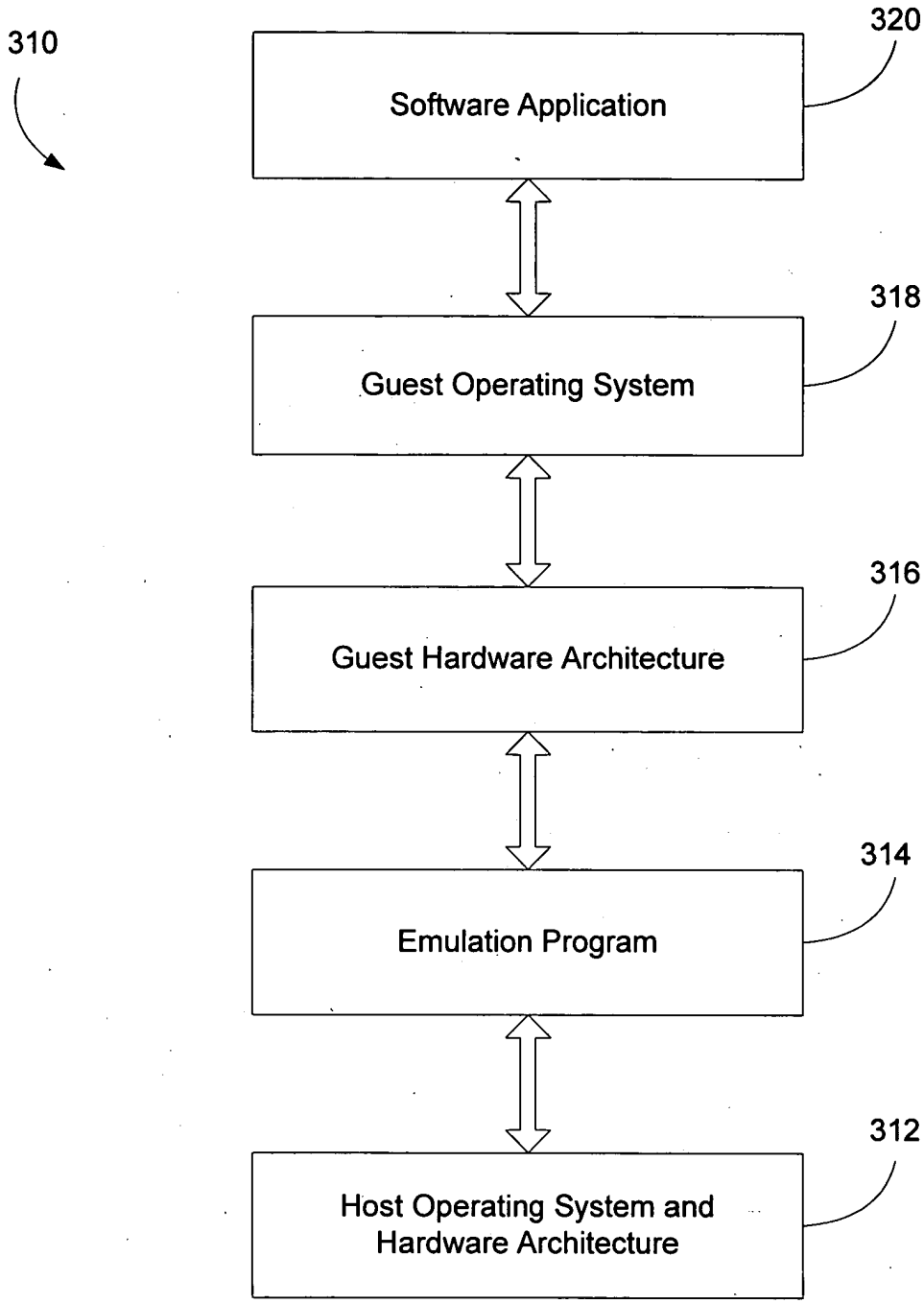
310
320
Software Application
318
Guest Operating System
316
Guest Hardware Architecture
314
Emulation Program
312
Host Operating System and Hardware Architecture

FIG. 3

22-32	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL		OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

FIG. 4

FIG. 5

Processor Mode	Default Virtualization Technique
Real mode (including SMM)	Emulation
Virtual 8086 mode	Direct Execution
Protected mode: Ring 0, 1 and 2	Emulation
Protected mode: Ring 3	Direct Execution

FIG. 6

Instruction	Virtualization Issue
PUSH CS	Exposes current ring level (which is encoded within the bottom two bits of CS)
PUSH SS	Exposes current ring level (which is encoded within the bottom two bits of SS)
MOV from SS	Exposes current ring level (which is encoded within the bottom two bits of SS)
CALLF	Exposes current ring level (which is encoded within the bottom two bits of the pushed CS)
SGDT	Non-trapping instruction exposes real (VMM) GDT
SLDT	Non-trapping instruction exposes real (VMM) LDT
SIDT	Non-trapping instruction exposes real (VMM) IDT
STR	Non-trapping instruction exposes real (VMM) TR

Docket No.: MSFT-2570 / 305147.01
App. No.: Not yet assigned Filed: Herewith
Title: SYSTEMS AND METHODS FOR USING SYNTHETIC INSTRUCTIONS IN A VIRTUAL MACHINE
Inventors: Eric Traut
Attorney: Richard W. Knight Phone: (215) 568-3100
Sheet 4 of 6

Instruction	Virtualization Issue
VERR	May return incorrect value because of modified DPL or descriptor type
VERW	May return incorrect value because of modified DPL or descriptor type
LAR	May return incorrect access rights because of modified DPL or descriptor type

FIG. 7

FIG. 8

Instruction	Virtualization Issue
CPUID	Non-trapping; no opportunity to return information that differs from the host processor.
PAUSE	Non-trapping; no opportunity to detect spin-lock condition on MP guest.

FIG. 9

Instruction	Virtualization Issue
PUSHF/PUSHFD	May push value within incorrect IF and IOPL fields.
POPF/POPFD	Value of IF and IOPL are ignored, even if guest CPL = 0.
IRET/IRETD	If returning to the same privilege level and same code selector (i.e. the CS on the stack is the same as the current CS), value of IF and IOPL are ignored, even if guest CPL = 0.

Docket No.: MSFT-2570 / 305147.01
App. No.: Not yet assigned Filed: Herewith
Title: SYSTEMS AND METHODS FOR USING SYNTHETIC INSTRUCTIONS IN A VIRTUAL MACHINE
Inventors: Eric Traut
Attorney: Richard W. Knight Phone: (215) 568-3100
Sheet 5 of 6

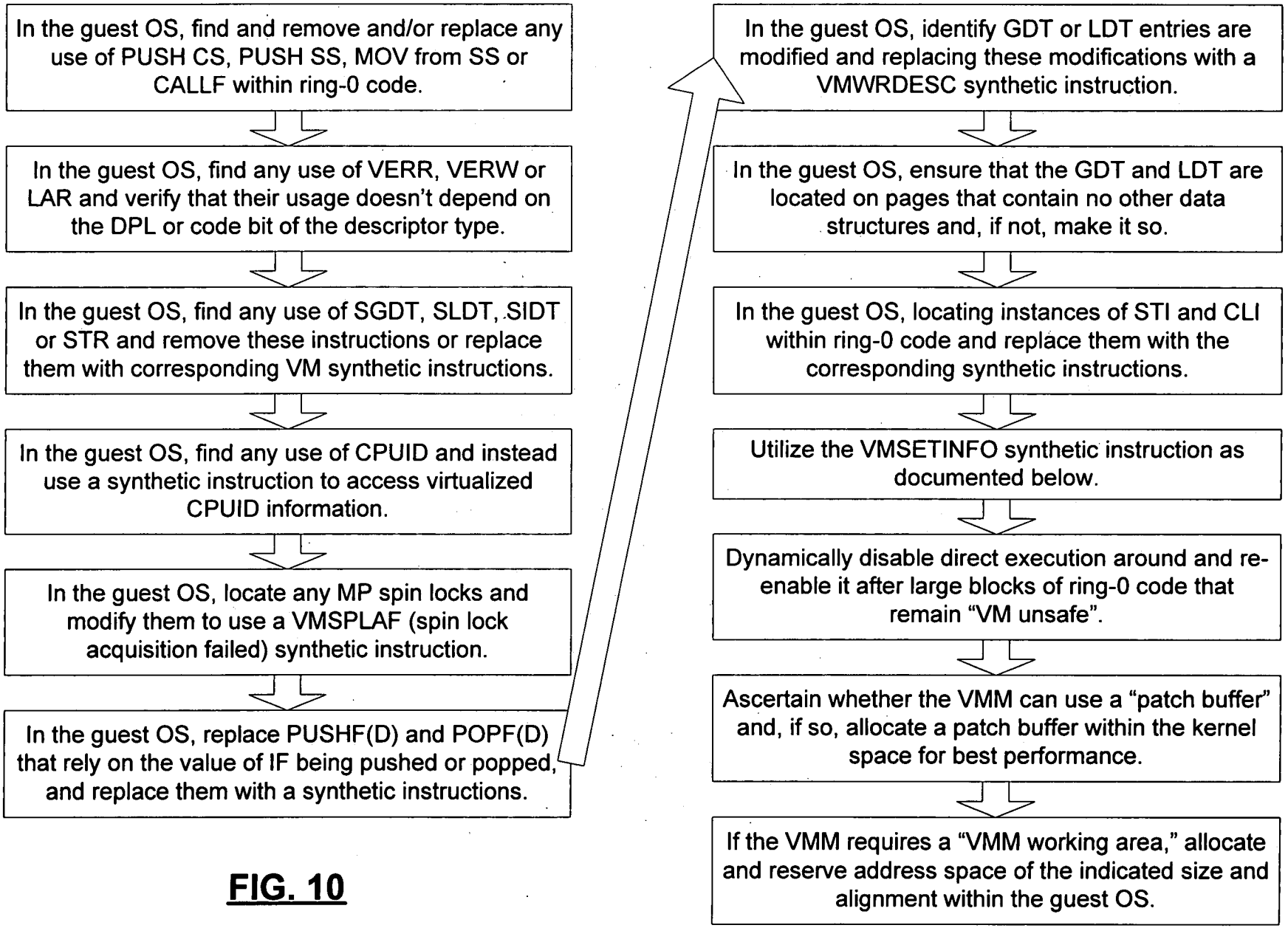
In the guest OS, find and remove and/or replace any use of PUSH CS, PUSH SS, MOV from SS or CALLF within ring-0 code.
In the guest OS, find any use of VERR, VERW or LAR and verify that their usage doesn't depend on the DPL or code bit of the descriptor type.
In the guest OS, find any use of SGDT, SLDT, SIDT or STR and remove these instructions or replace them with corresponding VM synthetic instructions.
In the guest OS, find any use of CPUID and instead use a synthetic instruction to access virtualized CPUID information.
In the guest OS, locate any MP spin locks and modify them to use a VMSPLAF (spin lock acquisition failed) synthetic instruction.
In the guest OS, replace PUSHF(D) and POPF(D) that rely on the value of IF being pushed or popped, and replace them with a synthetic instructions.
In the guest OS, identify GDT or LDT entries are modified and replacing these modifications with a VMWRDESC synthetic instruction.
In the guest OS, ensure that the GDT and LDT are located on pages that contain no other data structures and, if not, make it so.
In the guest OS, locating instances of STI and CLI within ring-0 code and replace them with the corresponding synthetic instructions.
Utilize the VMSETINFO synthetic instruction as documented below.
Dynamically disable direct execution around and re-enable it after large blocks of ring-0 code that remain "VM unsafe".
Ascertain whether the VMM can use a "patch buffer" and, if so, allocate a patch buffer within the kernel space for best performance.
If the VMM requires a "VMM working area," allocate and reserve address space of the indicated size and alignment within the guest OS.

FIG. 10